Support of Different Character Sets

- Character Sets that are Supported
  - Internal Character Set
  - External Character Sets
  - Character-Set Conversion
- Configuration File NATCONV.INI

This section describes how Natural supports different character sets.

The support of multiple languages with different character sets represents Natural’s first step towards internationalization. It can help you when using:

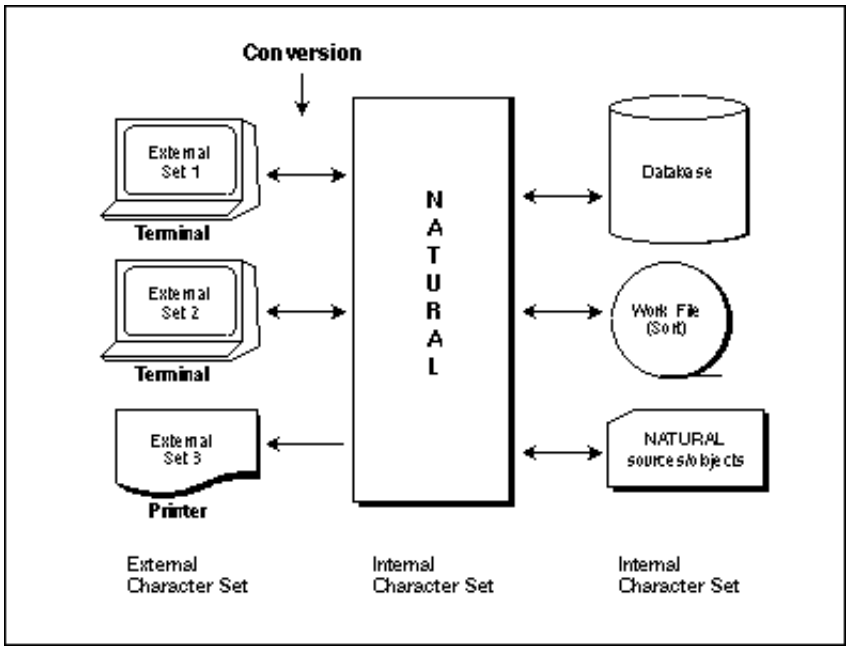
- terminals and printers with different character sets, all communicating with the same Natural environment;
- several Natural environments sharing one database and located on different platforms;
- language-specific characters in Natural identifiers, object names and library names;
- upper-/lower-case translation of language-specific characters;
- language-specific characters in an operand compared with a mask definition.

Character Sets that are Supported

Natural supports any single-byte character set that conforms to the ASCII character set in the lowest seven bits.

Natural distinguishes between an internal character and several external character sets; the internal character set is used by Natural itself.

As illustrated below, conversion between the internal and an external character set is performed after the input from a terminal and before the output to a terminal or printer. There is no conversion to an external character set available for work file I/Os, database I/Os and reading/writing of Natural objects.



Internal Character Set

By default, Natural uses the internal character set "ISO8859_1". If the default character set does not meet your requirements, you can choose either one of the predefined character sets provided by Natural or any other standard character set.

Note: Problems may occur if you run computers with different internal character sets sharing the same database or if you try to exchange data or programming objects between such computers.

External Character Sets

You are able to define an external character set for any terminal and printer.

For a terminal, the name of its character set is defined by the TCS entry in the terminal database, for example: ":TCS = usascii:".

You can also use the OpenVMS logical NATTCHARSET/the UNIX environment variable $NATTCHARSET which overrides all TCS settings.

If neither a TCS entry nor the logical NATTCHARSET/the environment variable $NATTCHARSET is defined, no conversion is performed during terminal I/O.

The character set name for a printer can be defined in the printer profile.

Character-Set Conversion

Natural performs the conversion between the internal character set and an external character set by using tables. For the most common character sets, these tables are predefined. You can modify existing tables and also define new ones.

The internal character set is used in the following situations:

- when checking identifiers,
- when translating from upper to lower case or vice versa,
- when classifying characters.

Identifier Checking

Natural checks identifiers (that is, user-defined variables in source programs), names of Natural objects and names of Natural libraries by using check tables. By modifying these tables, you can, for example, allow language-specific characters. In addition, you can redefine the characters "+", "#" and "&", which have a special meaning when used as the first character in variable names.

Upper-/Lower-Case Conversion

Natural performs upper-/lower-case conversion if you use one of the following:

- the %U terminal command,
- the AD = T field attribute,
- the EXAMINE TRANSLATE statement.

By modifying the translation tables, you can support a specific internal character set.

Configuration File NATCONV.INI

All check, translation and classification tables used by Natural to support a user-specific character set reside in the configuration file NATCONV.INI.

You can modify NATCONV.INI to support local or application-specific character sets.

In a standard application, NATCONV.INI need not and should not be modified, because this could lead to serious inconsistencies, in particular if Natural objects and database data are already present.

Modifications are necessary if you want to do any of the following:

- use an internal character set other than the default one,
- use a terminal or printer whose character set is still not supported by NATCONV.INI,
- allow or disallow the use of certain characters in identifiers,
- support local characters when evaluating the MASK option.

Any modifications of NATCONV.INI should be well considered and carefully performed, because otherwise, problems might occur that are difficult to locate.

NATCONV.INI is subdivided in sections and subsections. It contains five defined sections with the following names:

- CHARACTERSET-DEFINITION
- CHARACTERSET-TRANSLATION
- CASE-TRANSLATION
- IDENTIFIER-VALIDATION
- CHARACTER-CLASSIFICATION

CHARACTERSET-DEFINITION

This section defines the name of the internal character set. The default is ISO8859_1. If you choose a different character set, subsections for this character set must be contained in the following sections.

CHARACTERSET-TRANSLATION

This section contains the tables required for the conversion between the internal character set and external character sets. If you use, for example, a terminal with an entry in SAGtermcap of ":TCS = ASCII_GERMAN:" and if ISO8859_1 is used as internal character set, the following two subsections must be contained in this section:

- [ISO8859_1->ASCII_GERMAN]
- [ASCII_GERMAN->ISO8859_1]

CASE-TRANSLATION

This section contains the tables required for the conversion from upper to lower case. This conversion is done within the internal character set. If, for example, the internal character set is "ISO8859_5", the following two subsections must be contained in this section:

- [ISO8859_5->UPPER]
- [ISO8859_5->LOWER]

IDENTIFIER-VALIDATION

This section contains the tables required for the validation of identifiers, object names and library names. It contains a subsection for each defined internal character set. The special characters "#" (for non-database variables), "+" (for application-independent variables) and "&" (for dynamic source generation) can be redefined in this section. In addition, the set of valid first and subsequent characters for identifiers, object names and library names can be modified.

CHARACTER-CLASSIFICATION

This section contains the tables required for the classification of characters, which, for example, are used when evaluating the MASK option. It contains a subsection for each defined internal character set.

The section CHARACTERSET-DEFINITION as well as each subsection contain lines which describe how characters are to be converted and which characters are related with which attributes.

These lines are represented as follows:

```
line          ::=  key = value
key           ::=  name_key | range_key
name_key      ::=  keyword{ CHARS }
keyword       ::=  INTERNAL-CHARACTERSET | NON-DB-VARI | DYNAMIC-SOURCE |
                     GLOBAL-VARI | FIRST-CHAR | SUBSEQUENT-CHAR |
                     LIB-FIRST-CHAR | LIB-SUBSEQUENT-CHAR | ALTERNATE-CARET
                     ISASCII | ISALPHA | ISALNUM | ISDIGIT | ISXDIGIT |
                     ISLOWER | ISUPPER | ISCNTRL | ISPRINT | ISPUNCT |
                     ISGRAPH | ISSPACE
range_key     ::=  hexnum |
                     hexnum-hexnum
value         ::=  val {, val }
val           ::=  hexnum |
                     hexnum-hexnum
hexnum        ::=  xhexdigithexdigit |
                     Xhexdigithexdigit
```

Notes:
If "range_key" variable is specified on the left-hand side, the number of values specified on the right-hand side must correspond to the number of values specified in the key range, unless only one value is specified on the right-hand side, which is then assigned to each element of the key range.
When the "name_key" variable is specified on the left-hand side and the corresponding list of character codes does not fit in one line, it can be continued on the next line by specifying"name_key = " again. You must not start the lines with leading blanks or tabulators.

Examples of Valid Lines:

x00-x1f = x00	All characters between "x00" and "x1f" are converted to "x00".
x00-x7f = x00-x7f	All characters between "x00" and "x7f" are not converted.
x00-x08 = x00,x01-x07,x00	The characters "x00" and "x08" are converted to "x00" and characters between "x01" and "x07" are not converted.
ISALPHA = x41-x5a,x61-x7a,xc0-xd6,xd8 ISALPHA = xd9-xf6,xf8-xff	The attribute ISALPHA is assigned to all characters specified in these two lines.

Examples of Invalid Lines:

x41 = ’A’	All characters must be specified in hexadecimal format.
0x00-0x1f = 0x00	Hexadecimal values have to be specified in either of the following ways: x*digitdigit* X*digitdigit*
x00-x0f = x00,x01	The number of specified values does not correspond to the number of elements in the key range.

Sample NATCONV.INI File:

```
#************************************************************************
[CHARACTERSET-DEFINITION]

# defining the internal characterset for NATURAL

INTERNAL-CHARACTERSET = ISO8859_1

[CHARACTERSET-DEFINITION-END]
#************************************************************************

#************************************************************************
[CHARACTERSET-TRANSLATION]

# translation tables between internal and external characterset

#------------------------------------------------------------------------
[ISO8859_1->USASCII]
#translate ISO8859_1 to USASCII code

x00-x7F = x00-x7F
x80-xBF = x3F
xC0-xCF = x41,x41,x41,x41,x41,x41,x41,x43,x45,x45,x45,x45,x49,x49,x49,x49
xD0-xDF = x44,x4E,x4F,x4F,x4F,x4F,x4F,x3F,x4F,x55,x55,x55,x55,x59,x50,x73
xE0-xEF = x61,x61,x61,x61,x61,x61,x61,x63,x65,x65,x65,x65,x69,x69,x69,x69
xF0-xFF = x64,x6E,x6F,x6F,x6F,x6F,x6F,x3F,x6F,x75,x75,x75,x75,x79,x70,x59

[ISO8859_1->USASCII-END]
#------------------------------------------------------------------------

#------------------------------------------------------------------------
[USASCII->ISO8859_1]
#translate USASCII to ISO8859_1 code

x00-xFF = x00-xFF

[USASCII->ISO8859_1-END]
#------------------------------------------------------------------------

[CHARACTERSET-TRANSLATION-END]
#************************************************************************
```



```
#************************************************************************
[CASE-TRANSLATION]

# translation tables for lower/uppercase conversion of the internal
characterset
#------------------------------------------------------------------------
[ISO8859_1->LOWER]
#translate ISO8859_1 to lowercase

x00-x40 = x00-x40
x41-x5A = x61-x7A
x5B-xBF = x5B-xBF
xC0-xDE = xE0-xF6,xD7,xF8-xFE
xDF-xFF = xDF-xFF

[ISO8859_1->LOWER-END]
#------------------------------------------------------------------------

#------------------------------------------------------------------------
[ISO8859_1->UPPER]
#translate ISO8859_1 to uppercase

x00-x60 = x00-x60
x61-x7A = x41-x5A
x7B-xDF = x7B-xDF
xE0-xFF = xC0-xD6,xF7,xD8-xDE,xFF


[ISO8859_1->UPPER-END]
#------------------------------------------------------------------------

[CASE-TRANSLATION-END]
#************************************************************************
```

```
#************************************************************************
[IDENTIFIER-VALIDATION]

# tables for validation of identifiers

#------------------------------------------------------------------------
[ISO8859_1]

# special characters
#
# non DB variable
#                 '#'
NON-DB-VARI    = x23

# dynamic source generation
#                 '&'
DYNAMIC-SOURCE = x26

# global variable
#                 '+'
GLOBAL-VARI   = x2B

# valid first characters
#              '#' '&' '+' 'A'-'Z' 'a'-'z'
FIRST-CHAR   = x23,x26,x2B,x41-x5A,x61-x7A

# valid subsequent characters

#                   '#' '$' '&' '-' '/' '0'-'9'
SUBSEQUENT-CHAR   = x23,x24,x26,x2D,x2F,x30-x39

#                   '@' 'A'-'Z' '_' 'a'-'z'
SUBSEQUENT-CHAR-1 = x40,x41-x5A,x5F,x61-x7A

# valid first characters for library names
#                  'A'-'Z' 'a'-'z'
LIB-FIRST-CHAR   = x41-x5A,x61-x7A

# valid subsequent characters for library names
#                       '-' '0'-'9' 'A'-'Z' '_' 'a'-'z'
LIB-SUBSEQUENT-CHAR   = x2D,x30-x39,x41-x5A,x5F,x61-x7A

[ISO8859_1-END]
#------------------------------------------------------------------------

[IDENTIFIER-VALIDATION-END]
#************************************************************************
```



```
#************************************************************************
[CHARACTER-CLASSIFICATION]

# classification of characters according to the C library functions
#
#    isascii
#    isalpha
#    isalnum
#    isdigit
#    isxdigit
#    islower
#    isupper
#    iscntrl
#    isprint
#    ispunct
#    isgraph
#    isspace
#

#------------------------------------------------------------------------
[ISO8859_1]
# valid for ISO8859_1 as internal characterset

# ASCII characters
ISASCII = x00-x7F

# alphabetical characters
#         'A'-'Z' 'a'-'z' language specific alpha characters
#                         vvvvvvvvvvvvvvvvvvvvvv
ISALPHA = x41-x5A,x61-x7A,xC0-xD6,xD8-xF6,xF8-xFF

# alphanumeric characters
#         '0'-'9' 'A'-'Z' 'a'-'z' language specific alpha characters
#                                 vvvvvvvvvvvvvvvvvvvvvv
ISALNUM = x30-x39,x41-x5A,x61-x7A,xC0-xD6,xD8-xF6,xF8-xFF

# digit characters
#         '0'-'9'
ISDIGIT = x30-x39

# hexadecimal digit characters
#          '0'-'9' 'A'-'F' 'a'-'f'
ISXDIGIT = x30-x39,x41-x46,x61-x66
```

```
# lowercase characters
#         'a'-'z' language specific lowercase characters
#                 vvvvvvvvvvvvvvv
ISLOWER = x61-x7A,xDF-xF6,xF8-xFF


# uppercase characters
#         'A'-'Z' language specific uppercase characters
#                 vvvvvvvvvvvvvvv
ISUPPER = x41-x5A,xC0-xD6,xD8-xDE

# control characters
#
ISCNTRL = x00-x1F,x7F-x9F

# printable characters
#
ISPRINT = x20-x7E,xA0-xFF

# special characters
#
ISPUNCT = x21-x2F,x3A-x40,x5B-x60,x7B-x7E,xA1-xBF,xD7,xF7

# graphical characters
#
ISGRAPH = x21-x7E,xA1-xFF

# spacing characters
#
ISSPACE = x09-x0D,x20

[ISO8859_1-END]
#-------------------------------------------------------------------------

[CHARACTER-CLASSIFICATION-END]
#*************************************************************************
```

